Kappa Rule-Based Modelling in Synthetic Biology

Rule-based modelling, an alternative to traditional reaction-based modelling, allows us to intuitively specify biological interactions while abstracting from the underlying combinatorial complexity. One such rule-based modelling formalism is Kappa, which we introduce to readers in this chapter. We discuss the application of Kappa to three modelling scenarios in synthetic biology: a unidirectional switch based on nitrosylase induction in *Saccharomyces cerevisiae*, the repressilator in *Escherichia coli* formed from BioBrick parts, and a light-mediated extension to said repressilator developed by the University of Edinburgh team during iGEM 2010. The second and third scenarios in particular form a case-based introduction to the Kappa BioBrick Framework, allowing us to systematically address the modelling of devices and circuits based on BioBrick parts in Kappa. Through the use of these examples, we highlight the ease with which Kappa can model biological interactions both at the genetic and the protein-protein interaction level, resulting in detailed stochastic models accounting naturally for transcriptional and translational resource usage. We also hope to impart the intuitively modular nature of the modelling processes involved, supported by the introduction of visual representations of Kappa models. Concluding, we explore future endeavours aimed at making modelling of synthetic biology more user-friendly and accessible, taking advantage of the strengths of rule-based modelling in Kappa.

Keywords: rule-based modelling, Kappa, synthetic biology, switch, repressilator, BioBrick parts, modularity

Corresponding Author:

John Wilson-Kanamori j.r.wilson-kanamori@sms.ed.ac.uk

School of Informatics, University of Edinburgh

Author:

Vincent Danos v.danos@ed.ac.uk

School of Informatics, University of Edinburgh

Author:

Ty Thomson ty.thomson@gmail.com

Plectix BioSystems

Author:

Ricardo Honorato-Zimmer r.honorato@sms.ed.ac.uk

School of Informatics, University of Edinburgh

With thanks to Hristiana Pashkuleva for her help in sculpting and editing the manuscript.

1 Introduction

Rule-based modelling languages, including Kappa **(1)** and the BioNetGen Language **(2)**, have been the focus of attention in developing biological models that are concise, comprehensible, and easily extensible **(3)**. Such languages represent biological entities such as proteins, functional units of DNA, and mRNAs as agents. Agents are named sets of sites that can be used to hold state or bind and interact with other agents. Interactions are represented by rules in the form of precondition and effect, governed by an associated rate constant that determines how frequently the interaction occurs. Rules differ from reactions in that participating agents need not be fully specified in the precondition -- for example, phosphorylation at a particular site may occur independently from whether its neighbouring site is bound or not -- which means that a single rule may encompass any number of individual reactions. In this way, rule-based approaches alleviate the combinatorial explosion that results from molecular entities existing under multiple different conditions (for example, states of phosphorylation). The combination of different independent rule sets implicitly generates different overall systems, thus allowing modular development of subsystems and their composition into a conjoined whole.

The goal of this chapter is to provide an introduction to modelling in synthetic biology using the aforementioned Kappa rule-based modelling language. We do so by revisiting two of the best-known devices in synthetic biology, the toggle switch and the repressilator, in a series of case studies designed to gradually introduce the reader to the techniques involved. These techniques include the visual representations of protein interactions, and the Kappa BioBrick Framework for modelling BioBrick parts. We deliberately introduce these concepts and models in a non-specialist manner, in the hopes of reaching out to as wide an audience as possible.

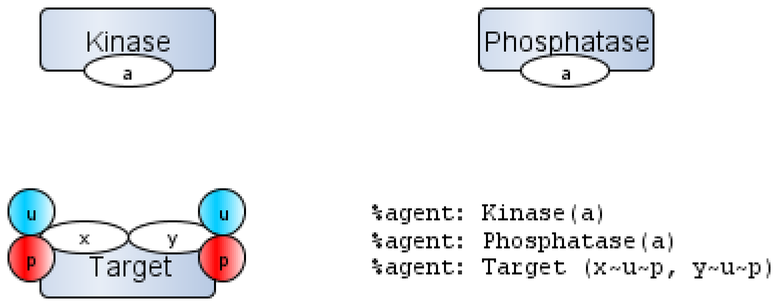
Although we adopt Kappa as our language of choice, the differences between it and the BioNetGen Language are minimal both in syntax and in implementation. However, one benefit of Kappa over BioNetGen is that Kappa tools utilise formal methods, such as causal summaries and reachability analysis, to aid information discovery in and debugging of large models.

Let us begin by introducing the key elements of Kappa in further detail.

An *agent* in Kappa is simply an entity with a name and a number of labelled *sites*. A site may hold internal *state*, typically used to denote a post-translational modification such as the agent's phosphorylation status. State may also be used to denote the agent's location in a model that takes into account different reaction compartments such as the cytosol and nucleus.

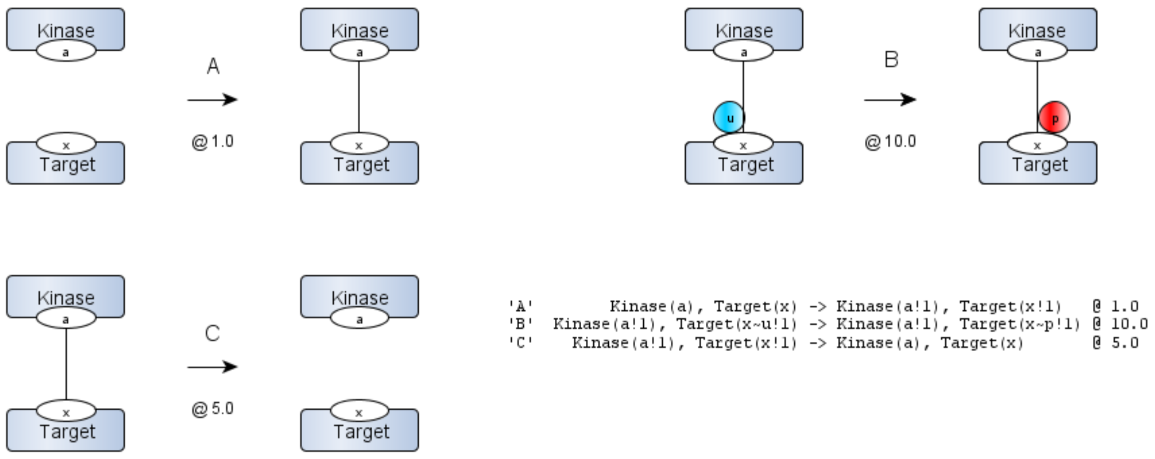
Agent interactions are described via *rules*. Rules often correspond to elementary mechanistic interactions such as the binding or unbinding of two agents, modification of the state of a site, or the creation or deletion of an agent. Complex rules can also describe combinations thereof.

As an introductory example, let us consider a basic kinase-phosphatase model. One has three agents (Figure 1.1): a Kinase, a Target with two sites 'x' and 'y' that may be independently phosphorylated, and a Phosphatase. A phosphorylation event may be simply described by three elementary actions (binding, modification, and unbinding) and their corresponding rules (Figure 1.2). In the corresponding textual notation (also shown in Figures 1.1 and 1.2), internal states are represented as '~u' (unphosphorylated) and '~p' (phosphorylated), and bindings as '!' with shared indices across agents indicating the two endpoints of a link. The left hand side of the rule describes the precondition that must be satisfied for the rule to apply. The right hand side describes its effect.



```
%agent: Kinase(a)
%agent: Phosphatase(a)
%agent: Target (x~u~p, y~u~p)
```

Figure 1.1: The agents involved in the basic kinase-phosphatase model: a Kinase with a single binding site 'a', a Phosphatase also with a single binding site 'a', and a Target with two binding sites 'x' and 'y' that may also switch state between phosphorylated ('~p') and unphosphorylated ('~u'). We show the textual code corresponding to each agent visualisation at bottom right.



```
'A'        Kinase(a), Target(x) -> Kinase(a!1), Target(x!1)   @ 1.0
'B'  Kinase(a!1), Target(x~u!1) -> Kinase(a!1), Target(x~p!1) @ 10.0
'C'    Kinase(a!1), Target(x!1) -> Kinase(a), Target(x)       @ 5.0
```

Figure 1.2: The three rules describing a phosphorylation in the kinase-phosphatase model. A) the Kinase binds its Target at one of the two sites (in this case 'x'); B) the Kinase phosphorylates the site at which it is bound; C) the Kinase dissociates (unbinds) from its target. Note the possibility that rule C might fire before rule B, thus not every binding event between a Kinase and a Target will result in a phosphorylation.

Note that not all sites of an agent need be present in a rule's precondition: the rules shown in Figure 1.2 never mention the Target's 'y' site. Likewise, even if a site is mentioned its internal state may be left unspecified: binding a Kinase to its Target does not take into account the fact that the Target may already be phosphorylated at that site. These are both examples of the "don't care, don't write" philosophy, where only the minimal information describing the triggering of a rule need be represented in the left hand side. This is what allows Kappa to alleviate the combinatorial explosion inherent in biological models.

The action of the Phosphatase, which works to counteract the Kinase, may be described using similar rules. The only difference between the two would then lie in the modification rule (Figure 1.2B), where the state of the Target changes from '~p' to '~u' rather than vice versa. Of course, this description would be a choice made by the modeller, and it would be entirely possible to model the Phosphatase (and the Kinase) differently but with a similar mechanistic effect. For example, we could design a 'smart' Phosphatase that only binds when a Target is already phosphorylated, or we could ensure that it never unbinds without dephosphorylating by combining the two effects into a single rule.

Every rule is associated with a *rate constant*, which controls the probability of the rule 'firing' during the simulation. At any given time in the simulation, the rule may apply to the *mixture* (the pool of interacting agents, including their binding configuration and internal state) multiple times according to how often the precondition holds. Each possible application has the same rate, hence the number of applications is multiplied by the rate of the rule to determine the rule's propensity (flux) at that point in time. The likelihood that the rule will fire next is proportional to this flux. Any obtained trajectory, of course, is but one realisation of a stochastic process that may differ when repeated. Manipulating the rate constants will influence these trajectories.

```
### Agents:
%agent: Kinase(a)
%agent: Phosphatase(a)
%agent: Target(x~u~p,y~u~p)


### Rules:

# Kinase action
'Kinase x binding'                 Kinase(a), Target(x) -> Kinase(a!1), Target(x!1)   @ 1.0
'Kinase x phosphorylation'  Kinase(a!1), Target(x~u!1) -> Kinase(a!1), Target(x~p!1) @ 10.0
'Kinase x unbinding'           Kinase(a!1), Target(x!1) -> Kinase(a), Target(x)       @ 5.0
'Kinase y binding'                 Kinase(a), Target(y) -> Kinase(a!1), Target(y!1)   @ 1.0
'Kinase y phosphorylation'  Kinase(a!1), Target(y~u!1) -> Kinase(a!1), Target(y~p!1) @ 10.0
'Kinase y unbinding'           Kinase(a!1), Target(y!1) -> Kinase(a), Target(y)       @ 5.0

# Phosphatase action
'Phtase x binding'                 Phosphatase(a), Target(x) -> Phosphatase(a!1), Target(x!1)   @ 1.0
'Phtase x phosphorylation'  Phosphatase(a!1), Target(x~p!1) -> Phosphatase(a!1), Target(x~u!1) @ 10.0
'Phtase x unbinding'           Phosphatase(a!1), Target(x!1) -> Phosphatase(a), Target(x)       @ 5.0
'Phtase y binding'                 Phosphatase(a), Target(y) -> Phosphatase(a!1), Target(y!1)   @ 1.0
'Phtase y phosphorylation'  Phosphatase(a!1), Target(y~p!1) -> Phosphatase(a!1), Target(y~u!1) @ 10.0
'Phtase y unbinding'           Phosphatase(a!1), Target(y!1) -> Phosphatase(a), Target(y)       @ 5.0


### Initial Conditions:
%init: 100  (Target(x~u,y~u))
%init: 10   (Kinase(a))
%init: 10   (Phosphatase(a))
```

Figure 1.3: The textual representation of the kinase-phosphatase model in Kappa. Agent definitions (Figure 1.1) are followed by the rules of the model (Figure 1.2) and the initial population. Simulation observables are not included, but are defined in a similar manner.

Perturbations form an integrated extension to the base Kappa language allowing the modeller to specify the effect of external influences on the model. For example, one may add and delete agents from the mixture conditioned on simulation time or other conditions pertaining to the state of the system.

Given a set of agents and rules as defined above, as well as appropriate *initial conditions*, one can then track user-defined *observables* in a stochastic *simulation*. The kinase-phosphatase example of a whole Kappa model that may be used for such a simulation is shown in Figure 1.3. The stochastic trajectories are obtained by using a rule-based variant of Gillespie's method **(4)** to simulate a continuous time Markov chain.

As well as this agent-centric view of system dynamics, automated tools exist to track causality and precedence in a model in a contrasting rule-centric view, for example to answer the question of what succession of events results in a fully phosphorylated Target. The idea behind such *stories* is to retain only the events in the causal lineage that contribute a net progression towards an event of interest (i.e. by removing causal loops).

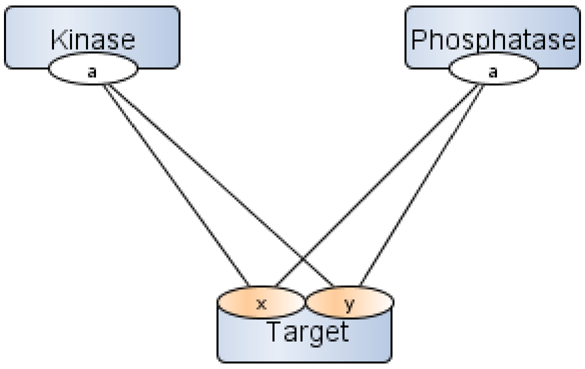


Figure 1.4: A contact graph for a simple kinase-phosphatase model. The three agents in the model are represented as nodes along with their sites, and each site is connected to the other sites it can bind to. If a site may be modified by this interaction, it is so indicated by a colour code (orange). Although simple enough, the associated rule set (three agents, twelve rules) already generates 38 non-isomorphic complexes (of which 36 contain the Target in various stages of binding and phosphorylation).

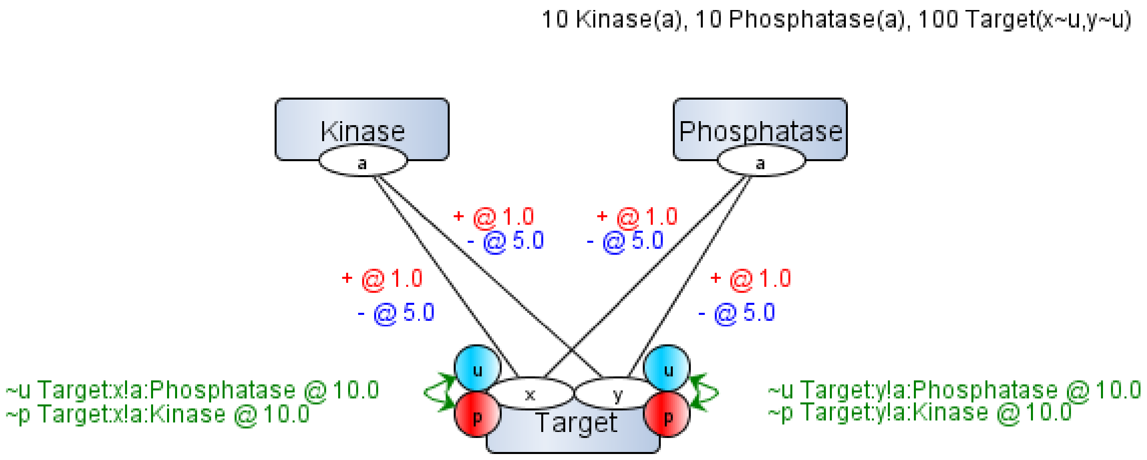

10 Kinase(a), 10 Phosphatase(a), 100 Target(x~u,y~u)
Kinase
a
Phosphatase
a
+ @ 1.0
- @ 5.0
+ @ 1.0
- @ 5.0
+ @ 1.0
- @ 5.0
+ @ 1.0
- @ 5.0
~u Target:x!a:Phosphatase @ 10.0
~p Target:x!a:Kinase @ 10.0
u
p
x
y
Target
u
p
~u Target:y!a:Phosphatase @ 10.0
~p Target:y!a:Kinase @ 10.0

Figure 1.5: An extension of the basic contact graph mapping the twelve rules (and their rates) of the model to the agents and their sites. States are shown explicitly, and the preconditions for their modification annotate the double-headed arrow between them. Agent interactions are annotated with the rate of binding (positive, in red) and unbinding (negative, in blue); as these are very basic binding and unbinding rules, no further annotations are needed. The entirety of this visualisation may be used to reconstruct the textual model as shown in Figure 1.3.

A useful overall view of the rule set is the *contact graph* (Figure 1.4), akin to a protein-protein interaction map. The contact graph is a graph whose nodes are the agents with their interfacing set of sites, and whose edges represent possible bindings between sites. Possible changes in state are indicated by a colour code upon the site in question.

The basic contact graph does not provide causal information, only possibilities, but we may extend it (Figure 1.5) by mapping to it the model rules categorised depending on their effect: binding and unbinding, creation, destruction, modification, and transport between different model compartments. Each edge in this visual formalism corresponds to one or more rules. In turn, each annotation on an edge describes the precondition (if any) and rate to one of these rules, such that said rule may be reconstructed wholly from the annotation. For example, two agents may bind with each other in two different rules dependent on their phosphorylation state; in the visualisation, this would be represented as a single edge between the two agents but annotated twice, once for each rule and attached rate.

Reversing this process thus allows us to recover the entirety of the model from the visualisation, minus simulation-specific observables. This extended visualisation is less useful when the model involves complex rules combining two or more of the above effects, however, and is currently defined manually (with plans for partial automation) in .graphml format (using the yED tool) from the underlying Kappa file.

To summarise the above: a Kappa model is a collection of agents (sets of sites that may hold state) and their rate-controlled interactions in the form of rules, which may be visualised concisely in the form of a contact graph. Given a set of initial conditions and observables, one may then execute this model to generate a stochastic simulation that tracks agent and mixture evolution over time, modelling external influences via perturbations and observing the causal properties of this evolution via stories.

A variety of tools exist for modelling in Kappa. Foremost amongst these is KaSim **(5)**, which may be used to simulate and analyse a Kappa model defined directly in textual form (as displayed in Figure 1.3). Under development for many years, the features embodied in KaSim are well-developed and user-tested. A recent alternative is LMS-Kappa **(6)**, an embedded domain-specific language written in Scala: users may write Kappa models using a Java-compatible object-oriented functional programming language with Java-like syntax to generate the rules. LMS-Kappa also eases the process of supplementing the core language with modular extensions (e.g. geometric constraints on the assembly of complexes) and makes the creation of Kappa models more accessible and more flexible from a programmer's perspective. Another recent effort is LBS-Kappa **(7)**, which attempts to capture the modularity inherent in common biological models to achieve concise representations, as an implementation of the Language for Biochemical Systems introduced elsewhere in this book.

With our introduction to Kappa now complete, we move on to some practical applications of the modelling language to synthetic biology. We highlight the manner in which models in Kappa may capture biological interactions at both the genetic and the protein-protein interaction levels through a series of case studies:

- a unidirectional switch based on nitrosylase induction in *Saccharomyces cerevisiae*, building on the kinase-phosphatase model introduced in this section.
- the repressilator in *Escherichia coli* formed from BioBrick parts, introducing the Kappa BioBrick Framework.
- a light-mediated extension to said repressilator developed by the University of Edinburgh team during iGEM 2010, to demonstrate how to build on the Kappa BioBrick Framework to create a model combining both protein and genetic interactions.

All models, including the introductory model described in this section, are available at RuleBase (http://rulebase.org), an online repository for rule-based models; the models from this chapter are located at http://rulebase.org/users/884. Prospective modellers will also find other examples of biologically interesting models there.

2 Modelling a Synthetic Switch

We begin exploring the role of the Kappa modelling language in synthetic biology by applying it to a real-life example of a synthetic switch. The design of this example flows naturally from the kinase-phosphatase model described in Section 1, and follows the idea of controlling devices based on Goldbeter-Koshland kinetics **(8)** known to possess ultrasensitive toggle-like properties in specific parameter ranges. Concretely, we use inducible promoters to control the two 'arms' of the Goldbeter-Koshland loop, represented previously by Kinase and Phosphatase and in the example to follow as TRXh5 and GSNO.

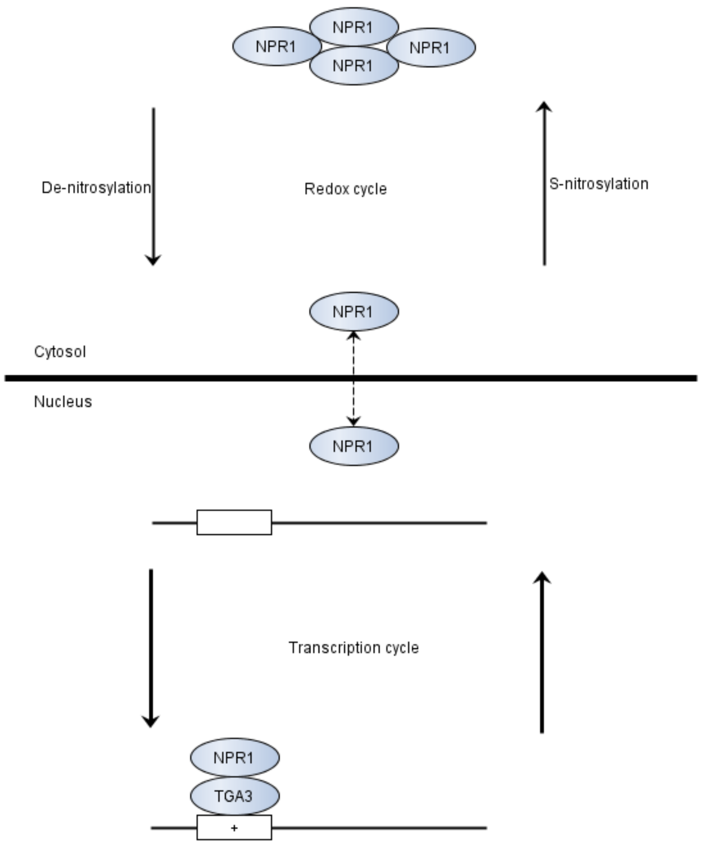


Figure 2.1: Under resting conditions, the S-nitrosylation of NPR1 by GSNO promotes formation of the NPR1 oligomer. Pathogen recognition causes ambient changes to cellular redox potential, which promotes monomer release. Nucleic monomeric NPR1 interacts and forms a complex with transcription factors at target promoters to activate them.

The constitutively expressed NPR1 transcription cofactor is an integral coordinator **(9)** of the multi-layered cellular defence system in *Arabidopsis* (Figure 2.1). In resting cells, it is subject to *S*-nitrosylation at a known amino acid (Cys156) by the GSNO protein acting to promote the assembly of the NPR1 oligomer. Modification to the intracellular redox environment by invasive pathogens, and the subsequent action of the enzymes TRXh5 and NTRA, releases the NPR1 monomer instead. Monomeric NPR1 interacts with the TGA subclass of transcription factors essential for activating immune defence genes, thus stimulating the cell's genetic response to pathogen invasion.

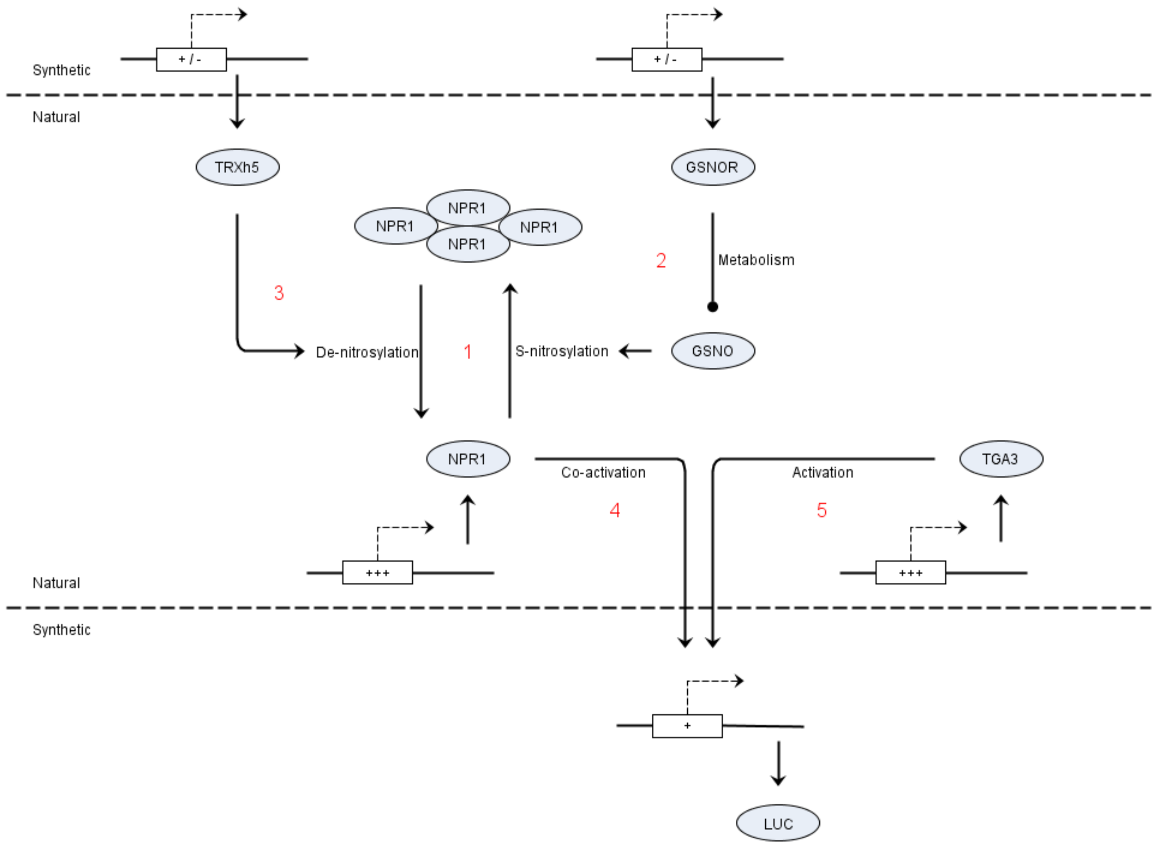


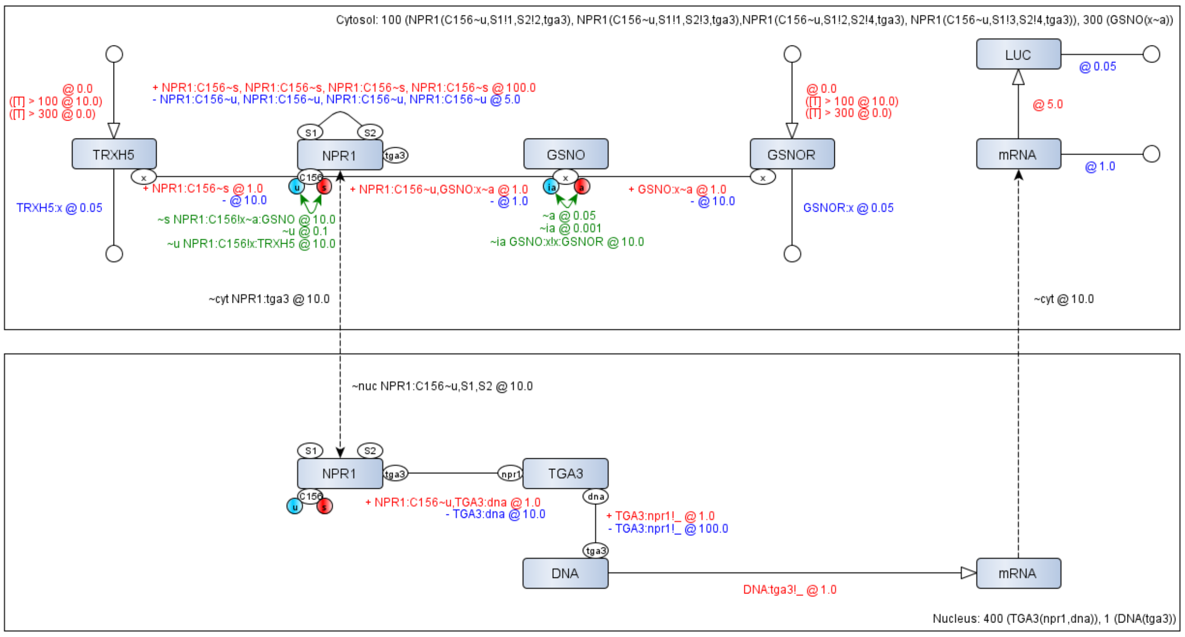
Figure 2.2: A schematic of the unidirectional synthetic protein switch. The labels on the promoters indicate inducible by an outside chemical compound (+/-), constitutively expressed (+++), and activated as pathway output (+) respectively. Although the system itself is natural (albeit transcribed from *Arabidopsis* to yeast, the inducible promoters and the reporter are synthetic. Note the structural similarities to the basic kinase-phosphatase model -- a target (NPR1) that switching between conformations (oligomeric to monomeric) under the influence of TRXh5 (forwards) and GSNO (backwards). 1) Under resting conditions, GSNO promotes oligomer formation of constitutively expressed NPR1. 2) Selectively activating GSNOR increases the rate at which GSNO is metabolised, thereby reducing S-nitrosylation of NPR1 and indirectly limiting oligomer formation. 3) Activation of TRXh5 (together with NTRA) de-nitrosylates the NPR1 oligomer and promotes monomer release. 4) Monomeric NPR1 interacts with constitutively expressed TGA3, readying it for activity as a transcription factor. 5) This then activates the expression of a LUC reporter gene.

Investigating this plant immune cascade allows us to identify system function influencing the outcome of defence signalling. In particular, much of the processes surrounding NPR1 function are not fully understood. Work by John Moore at the University of Edinburgh **(10)** approaches this problem by creating a synthetic protein circuit based on a theoretical interpretation of plant immunity, using a *Saccharomyces cerevisiae* yeast chassis engineered to be amenable to redox manipulation.

By designing a synthetic circuit (Figure 2.2) with selectively inducible inputs in the form of TRXh5 and the GSNO reductase GSNOR, we specifically promote the reduction of NPR1 to its monomeric form. This creates a unidirectional 'on' switch with the standard luciferase reporter gene as the circuit output. It is not a true toggle switch as described by Gardner et al. **(11)**, since there is no inducible mechanism for directly turning the circuit 'off' again, but rather a gradual return to the pre-induction state as the inducible inputs filter out of the system.

We then develop a Kappa model to provide a mathematical representation of the system and to facilitate further investigation via model perturbations to supplement experimental analysis. This model is visualised in Figure 2.3. The model visualisations described in Section 1 are extended further here to incorporate compartment information in a rudimentary fashion (again, modelled as a location site holding state either '~cytosol' or '~nucleus'). The compartments are supplemented with the initial populations of the agents present.

We begin our modelling case study by assuming the constitutive expression of NPR1, GSNO, and TGA3 proteins. Initially in the cytosolic compartment, the C156 site of the NPR1 agent favours its *S*-nitrosylated '~s' state due to the action of GSNO. GSNO and NPR1 can bind if NPR1 is de-nitrosylated (its 'C156' site is in its '~u' state) and GSNO is active (its 'x' site is in its '~a' state); once bound, another rule allows NPR1 to *S*-nitrosylate by switching state from '~u' to '~s' on its 'C156' site. This ensures that NPR1 retains its oligomeric form, represented by four NPR1 agents forming a tetrameric ring. In turn, this prevents NPR1 from leaving the cytosol (since our transport rules state that nucleic transport of NPR1 can only occur when it is de-nitrosylated and its 'S1' and 'S2' sites are unbound, i.e. it is in monomeric form) and activating the downstream reporter in the nuclear compartment. These rules are fully visualised in Figure 2.3.



Cytosol: 100 (NPR1(C156~u,S1!1,S2!2,tga3), NPR1(C156~u,S1!1,S2!3,tga3),NPR1(C156~u,S1!2,S2!4,tga3), NPR1(C156~u,S1!3,S2!4,tga3)), 300 (GSNO(x~a))
@ 0.0
([T] > 100 @ 10.0)
([T] > 300 @ 0.0)
+ NPR1:C156~s, NPR1:C156~s, NPR1:C156~s, NPR1:C156~s @ 100.0
- NPR1:C156~u, NPR1:C156~u, NPR1:C156~u, NPR1:C156~u @ 5.0
S1
S2
TRXH5
NPR1
tga3
GSNO
GSNOR
LUC
@ 0.05
@ 5.0
mRNA
@ 1.0
x
C156
u
s
+ NPR1:C156~s @ 1.0
- @ 10.0
TRXH5:x @ 0.05
~s NPR1:C156!x~a:GSNO @ 10.0
~u @ 0.1
~u NPR1:C156!x:TRXH5 @ 10.0
+ NPR1:C156~u,GSNO:x~a @ 1.0
- @ 1.0
ia
a
~a @ 0.05
~ia @ 0.001
~ia GSNO:x!x:GSNOR @ 10.0
+ GSNO:x~a @ 1.0
- @ 10.0
@ 0.0
([T] > 100 @ 10.0)
([T] > 300 @ 0.0)
GSNOR:x @ 0.05
~cyt NPR1:tga3 @ 10.0
~cyt @ 10.0
~nuc NPR1:C156~u,S1,S2 @ 10.0
S1
S2
NPR1
tga3
npr1
TGA3
C156
u
s
+ NPR1:C156~u,TGA3:dna @ 1.0
- TGA3:dna @ 10.0
dna
+ TGA3:npr1!_ @ 1.0
- TGA3:npr1!_ @ 100.0
tga3
DNA
mRNA
DNA:tga3!_ @ 1.0
Nucleus: 400 (TGA3(npr1,dna)), 1 (DNA(tga3))

Figure 2.3: Visualising the Kappa model of the synthetic immune pathway. GSNO, NPR1, and TGA3 are assumed to be constitutively expressed at rest, with GSNO promoting formation of the NPR1 oligomer. In addition to the visualisation concepts described previously, we introduce the notion of compartments and transport between them (indicated by the dashed arrows), creation and degradation of agents (indicated by source and sink nodes) and the perturbation language (indicated by the bracketed annotations on the edges leading from the source nodes). At simulation time T=100, GSNOR and TRXh5 are inserted into the system via perturbation, de-nitrosylating NPR1 and promoting its monomeric form instead. As described in the text, monomeric NPR1 then translocates to the nucleus, combining with TGA3 to activate the expression of the reporter gene. The transcribed mRNA is transported back to the cytosol, producing LUC output.

The implementation illustrates an important adage: the information contained in the model is directly related to what is known of the system and the desired level of abstraction. For example, we make the choice in this example to model the NPR1 oligomer as a tetrameric ring of four separate agents rather than simply a state (oligomer vs. monomer) of the individual agent. This choice has further consequences on system dynamics, as we shall see shortly.

TRXh5 and GSNOR agents do not exist in the initial population of the cytosol, but are introduced from simulation time 100 via a model perturbation substituting for the expression of the relevant synthetic construct (the perturbation is represented in parentheses on the creation edge in Figure 2.3).

This influx is turned off again at simulation time 300, by which time enough TRXh5 and GSNOR are introduced to toggle the system switch. TRXh5 directly de-nitrosylates NPR1, switching the state of the 'C156' site to '~u', whilst GSNOR inactivates GSNO and prevents it from acting on NPR1 to reverse this change; the second effect takes place over a slightly longer timescale than the first. Cumulatively, the two proteins stimulate the release of the monomeric form of NPR1, which we choose to model as the oligomeric NPR1 breaking up into its constituent parts once all four NPR1 agents are de-nitrosylated. The system thus behaves explosively as a large number of NPR1 monomers are released in a short time; in contrast, if we had chosen to model the conformation of NPR1 as a state as described previously, then we would have seen a more gradual change. Again, although mathematical models are often helpful in describing hitherto unknown behaviour, the system can only behave according to the assumptions we make when we model it.

Monomeric NPR1 is free to transport to the nucleic compartment, where it interacts with the TGA3 transcription factor to activate the reporter gene. The mRNA thus produced is transported back to the cytosol, where in the final stage of the model cascade it is translated into the LUC reporter protein -- the output of the synthetic circuit.

Switching off TRXh5 and GSNOR induction causes the system to return slowly to its pre-induction state as the two protein populations are filtered out by constitutive degradation. NPR1 returns to its oligomeric form, spurred by the *S*-nitrosylating action of reactivated GSNO, and the reporter gene is turned off. Hence we have a unidirectional switch structure controllable by the induction described previously.

In vivo studies of the synthetic circuit have determined that it takes two hours for GSNOR/TRXh5 to appear once induced, four to five hours for the threshold level of de-nitrosylated NPR1 to be reached, 90 minutes for the cytosolic oligomer to convert to a nucleic monomer, and a further two to three hours for LUC protein creation. Unfortunately these abstract observations are insufficient in themselves to fully specify the kinetic rates that determine model dynamics. Hence, as a first step we simply guess at these rate constants in the actual model, although simultaneously we maintain the initial protein populations as faithful to observed biology as possible.

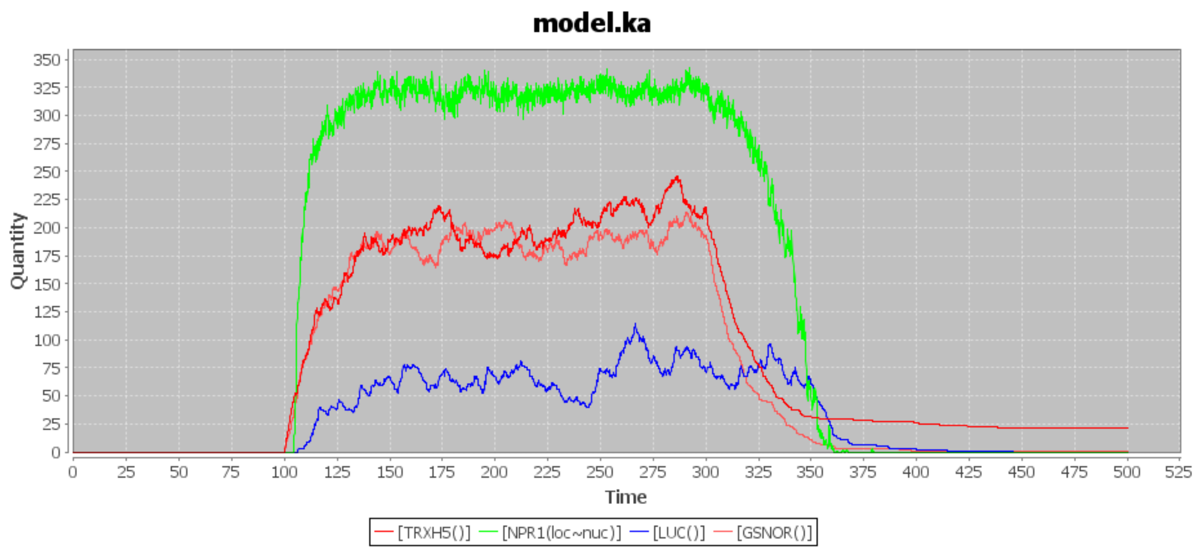


Figure 2.4: Stochastic simulation of the NPR1 oligomer-to-monomer switch modelled in Kappa. Plot demonstrating that nuclear-localised monomeric NPR1 and subsequent reporter gene expression is dependent on induction of GSNOR and TRXh5. GSNOR and TRXh5 are switched on at simulation time T=100 and off at T=300. Units (time and quantity) are arbitrary.

As a result, at this stage simulations of the model (Figure 2.4) display highly qualitative dynamics that are useful for tracking causality but not for reproducing the *in vivo* temporal behaviours described above. Given more careful characterisation and analysis, our model is poised to make pertinent observations regarding the sensitivity of the NPR1 oligomer-monomer switch to the concentrations of TRXh5 and GSNO in the system, as well as the responsiveness (how long it takes to complete stimulation response) and strength (the proportion of oligomeric NPR1 converted to monomeric form) of the switch. For the time being, it serves as an example of how we may model simple synthetic circuits, and in particular the protein interactions that may compose such circuits, in Kappa.

Now that we have demonstrated how to proceed from toy examples (kinase-phosphatase) to full-fledged biological models, our next question is how to tackle structured synthetic biology based on BioBrick parts.

3 The Kappa BioBrick Framework

Modular methodologies for modelling structured synthetic biological systems, based on the BioBrick standard **(12)** and formalised by systems of ordinary differential equations, were first explored by Marchisio and Stelling in 2008 **(13)**. Other tools such as TinkerCell **(14)** facilitate modellers looking to incorporate important principles such as stochasticity and analysis paradigms including parameter scanning into the modelling of BioBrick parts. We introduce in this section such a methodology, designed to assist the modelling in Kappa of circuits based on BioBrick parts.

The Kappa BioBrick Framework, originally laid down by the third author in 2009, differs from the above by incorporating the advantages of rule-based modelling with the description of dynamic BioBrick parts. Given a specification of a device constructed from BioBrick parts, it provides rules describing how these gene constructs are processed by the transcription and translation machinery of the cell (RNA polymerases and ribosomes). Such a framework allows the modular formalisation of individual functional units within the system and their composition into more complicated devices and systems. It also meshes well with incremental strategies for modelling synthetic biological systems. Finally, the structure of the Kappa BioBrick Framework corresponds with efforts to standardise the characterisation of BioBrick parts, utilising measures such as Polymerases Per Second (PoPS, the rate of transcription defined as the number of times that an RNA polymerase molecule passes a specific point on DNA per second) or Ribosomal Initiations Per Second (RiPS, the level of translation as the number of ribosome molecules that pass a point on mRNA each second).

Our approach is similar to later work by Marchisio et al. **(15)** regarding the use of the BioNetGen language in a framework for the design of complex eukaryotic gene circuits. Unlike this, the Kappa BioBrick Framework does not specifically cater for synthetic biology based on mRNA regulation using small RNAs, or eukaryotic issues such as compartmentalisation or RNA interference, although these may of course be added by the modeller. An additional difference is that Marchisio et al. utilise the Model Description Language to provide an interface between modules; the Kappa BioBrick Framework relies instead on the inherent compositionality of the rules in Kappa.

We provide further details of a preliminary implementation of the Kappa BioBrick Framework in the conclusion of this chapter. For now, we begin describing the Kappa BioBrick Framework by considering the four functional categories of BioBrick parts: promoters, coding sequences, ribosome binding sites (RBS), and terminators.

A BioBrick part in the framework consists of one or more DNA agents connected in a chain and tagged with a unique identifier, for example adopted from the Registry of Standard Biological Parts (http://partsregistry.org/). Each BioBrick part also has an RNA representation defined in a similar manner. Given a set of BioBrick parts, the framework will automatically generate these DNA and RNA agents, along with RNA polymerases (RNAP) and Ribosomes involved in transcription and translation, and placeholder Transcription Factor and Protein agents to be manually refined by the user (Figure 3.1). Organising the agents in this manner allows us to represent the transcriptional and translational interactions modularly, without placing any limitations on protein behaviour (which may vary a great deal more) beyond activity as a transcription factor.

The framework also generates a concise and complete set of rules (with associated kinetic rates) necessary to describe the activity of these BioBrick parts in an idealised space-homogeneous chassis. Once the rules for a virtual part have been established, it can be composed with other virtual parts in a modular manner analogous to the use of actual BioBrick parts in synthetic biology. The paragraphs that follow may be thought of as a recipe with which to create a set of Kappa rules for a certain part.

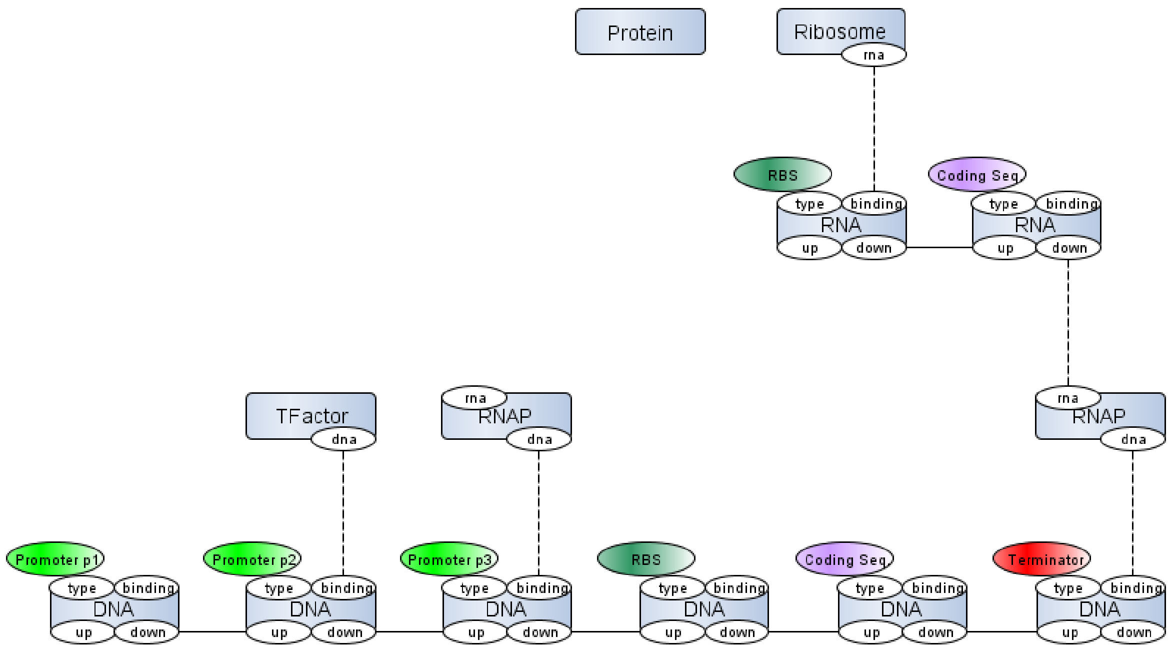


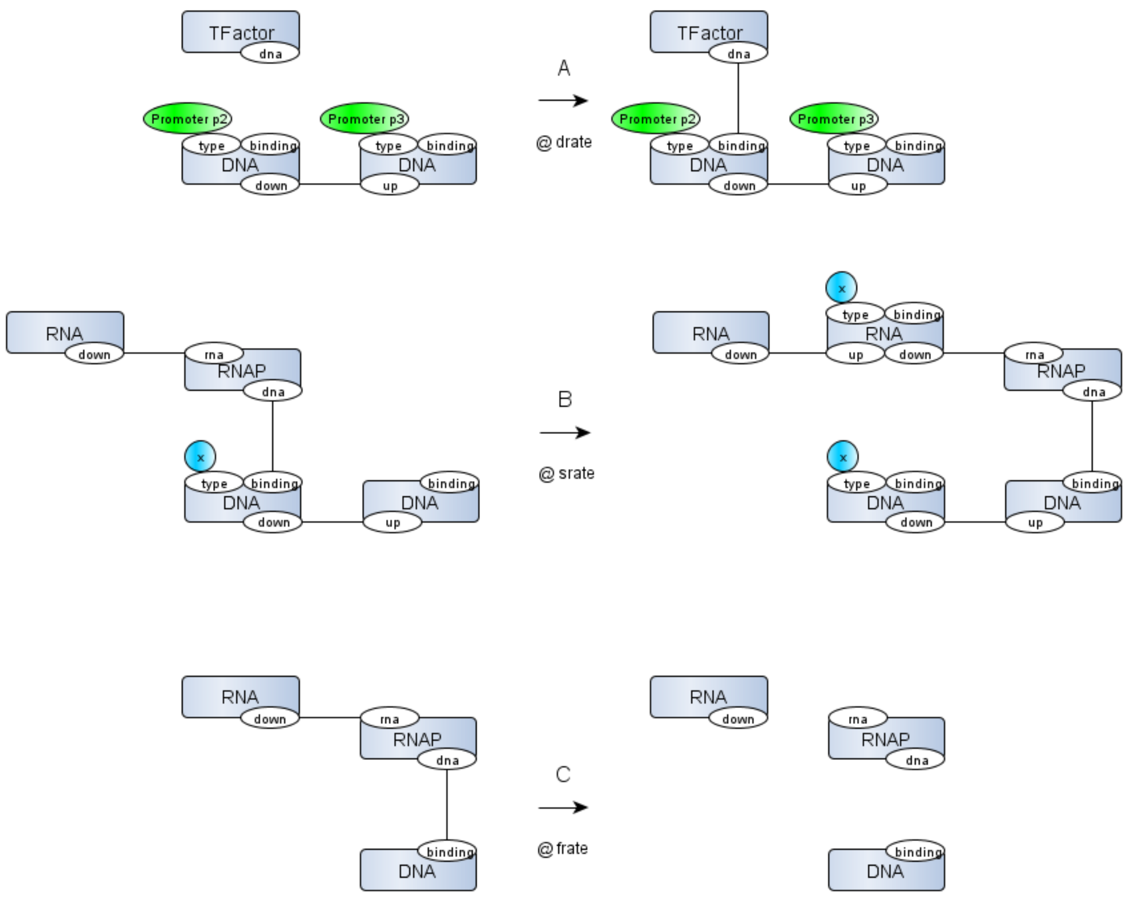
Figure 3.1: The agents involved in the Kappa BioBrick Framework. At bottom, a chain of DNA agents (linked by their upstream and downstream sites) representing the BioBrick parts; each part would have a unique identifier contained as a state to the 'type' site. mRNA is also represented as such a chain, transcribed from the DNA by RNAP and subject to translation by Ribosomes. The TFactor and Protein agents are placeholders automatically generated by the framework, and must be manually refined into the corresponding protein.

The crux of the framework lies in three basic rule templates: 'docking', 'sliding', and 'falloff', shown in Figure 3.2. Every rule generated by the framework, bar the universal RNA degradation rule, is instantiated from one of these templates. At the transcriptional level:

- The 'docking' rule template (Figure 3.2A) defines the mechanism of transcription factor and RNA polymerase binding to BioBrick promoter parts. The regulatory effect of transcription factors on BioBrick promoters are simply refinements of these 'docking' rules (Figure 3.3).
- All BioBrick DNA agents require associated rules that describe the transcription of the part from DNA to RNA caused by RNA polymerase; these are the 'sliding' rules (Figure 3.2B).
- 'Falloff' rules (Figure 3.2C) deal with the detachment of RNA polymerase and transcription factors from the chain of DNA agents representing BioBrick parts. Although this may in theory happen on any BioBrick part, BioBrick terminators have a higher falloff rate due to their function in preventing further transcription downstream.

Similarly at the translational level, ribosomes may 'dock' at ribosome binding sites, may 'slide' across protein coding sequences to translate the appropriate protein, and may 'falloff' from the RNA chain. The framework also describes the potential degradation of RNA agents at a uniform rate, unlike DNA agents which are assumed to be immutable barring user-defined rules.

The rate constants associated with these framework rules correspond to values that would ideally be known to a rich database of synthetic BioBrick parts. For example, the effective rates of 'sliding' rules would correlate with measurements of PoPS for transcriptional activity and RiPS for translational activity. 'Docking' rules would relate to such measures as promoter and RBS strength: transcription factor affinity, RNAP attraction as a function of transcription factor binding, and ribosome affinity. The kinetic rate for 'falloff' rules is determined by such factors as RNAP error rate and terminator efficiency.



```
'A (Docking)' DNA(type~p2,binding,down!1), DNA(type~p3,binding,up!1), TFactor(dna) ->
              DNA(type~p2,binding!2,down!1), DNA(type~p3,binding,up!1), TFactor(dna!2)    @ drate
'B (Sliding)' DNA(type~x,binding!2,down!1), DNA(binding,up!1), RNAP(dna!2,rna!3), RNA(down!3) ->
              DNA(type~x,binding,down!1), DNA(binding!2,up!1), RNAP(dna!2,rna!3),
              RNA(type~x,binding,down!3,up!4), RNA(down!4)                                @ srate
'C (Falloff)' DNA(binding!1), RNAP(dna!1,rna!2), RNA(down!2) ->
              DNA(binding), RNAP(dna,rna), RNA(down)                                      @ frate
```

Figure 3.2: Instances of the trinity of rules in the Kappa BioBrick Framework: 'docking', 'sliding', 'falloff'. A) The 'docking' rule illustrates 'transcription factor binding' rule for a promoter from Table 3.1: the binding of a transcription factor to the correct binding region on the promoter BioBrick part, given that there is no RNAP present (note that the binding region on the downstream DNA agent is free). B) The 'sliding' rule illustrates a 'translation' rule for a protein coding sequence: an RNAP agent moving along the DNA chain and transcribing the appropriate RNA as it does so. C) The 'falloff' rule illustrates both universal RNAP falloff and the falloff for a terminator: the dissociation of RNAP from an unspecified DNA part (the terminator falloff rule would have a higher rate constant than the universal rule), and the simultaneous release of the constructed RNA chain (there may be any number of RNA agents upstream of the one shown).

It must be clearly stated that the above rules do not take into account the actions of the proteins after they are synthesised or any pathways that they may be involved in, bar their possible effect as a transcription factor. Such considerations (for example, protein degradation or kinase activity) are up to the individual modeller to incorporate separately in addition to rules generated by the framework. An example of such an effort is provided in Section 4 to follow. Of course, at this point the advantages of modelling in Kappa that we have already explored in Section 2 apply.

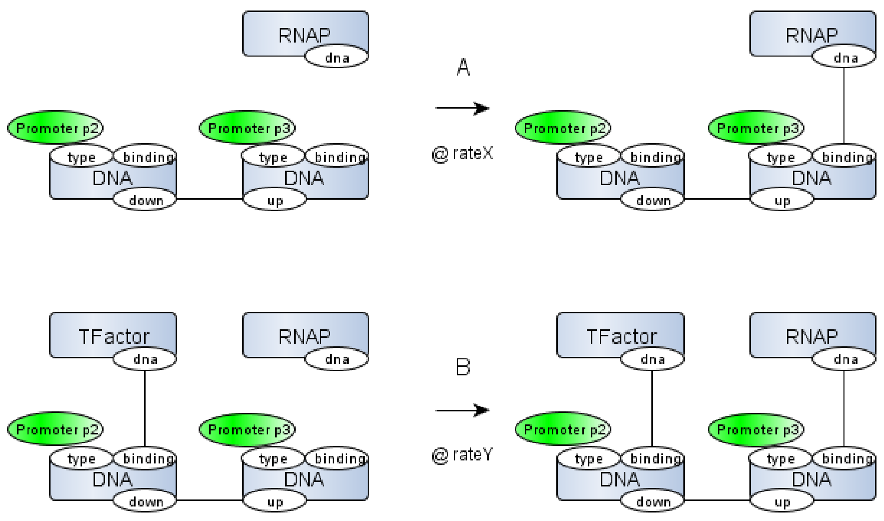


Figure 3.3: Adapting the 'docking' rules to model transcription factor regulation upon promoter activity. A) The binding of RNAP to a promoter whose regulatory site is free, governed by a rate constant rateX. B) The binding of RNAP to a promoter whose regulatory site is bound to a generic transcription factor, similarly governed by a rate constant rateY. The transcription factor is an activator in rateX is negligible and rateY is significant; conversely, if rateX is significant but rateY negligible, then the transcription factor is a repressor. By manipulating the exact values of rateX and rateY we are able to define leaky promoters or promoters of varying strengths. Finally, we can represent promoters with multiple regulatory binding sites, for example cooperative regulation, simply by elongating the promoter part (increasing the number of DNA agents involved) and specifying the effect of multiple transcription factors upon RNAP attraction.

	Transcription	Translation
Promoter	Transcription factor binding (*)	
	Transcription factor unbinding (*)	
	RNA polymerase binding (*)	
	Transcription initiation	
	Transcription readthrough	
Ribosome Binding Site	Transcription	Ribosome binding
Protein Coding Sequence	Transcription	Translation initiation
		Translation (*)
Terminator	Transcription termination	
	Transcription readthrough	
Other	RNA polymerase falloff	Ribosome falloff
		RNA degradation

Table 3.1: The rules generated by the Kappa BioBrick Framework at both transcriptional and translational levels. 'Docking' rules are denoted in red, 'sliding' rules in blue, and 'falloff' rules in green. The marked promoter rules need to be manually refined according to promoter structure (repressor or activator, number of transcription factor binding sites); the protein coding sequence translation rule similarly requires user input of the specific protein agent created.

The rules of the Kappa BioBrick Framework may thus be categorised along three dimensions: the BioBrick part they are associated with, whether they are involved at the transcriptional or translational level of cellular machinery, and which of the three templates they are based on (Table 3.1). To illustrate them further, we now present a model of the hallowed Elowitz repressilator created from BioBrick parts **(16)**.

3.1 The Repressilator in the Kappa BioBrick Framework

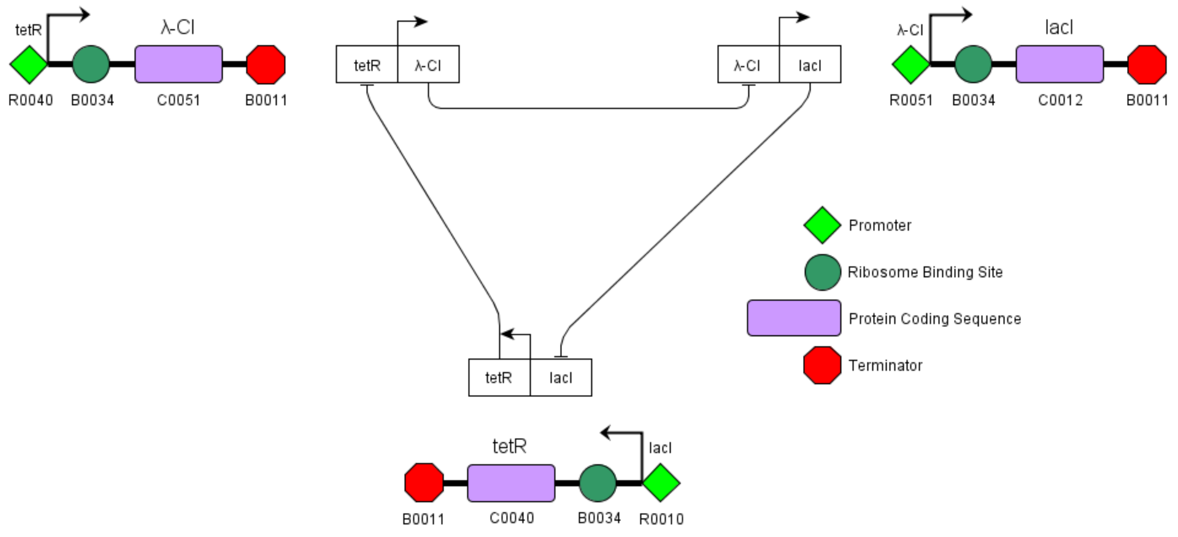


Figure 3.4: The Elowitz repressilator constructed from BioBrick parts, with each repressor inhibiting production of the next repressor in the loop through the action of the appropriate promoter. Each part may be represented as a modular component (a set of agents and associated rules) in the Kappa BioBrick Framework, and the composition of a number of these parts creates a full circuit as shown. The central representation hides the RBS and terminator components of the BioBrick device, since we reuse the same parts throughout the model.

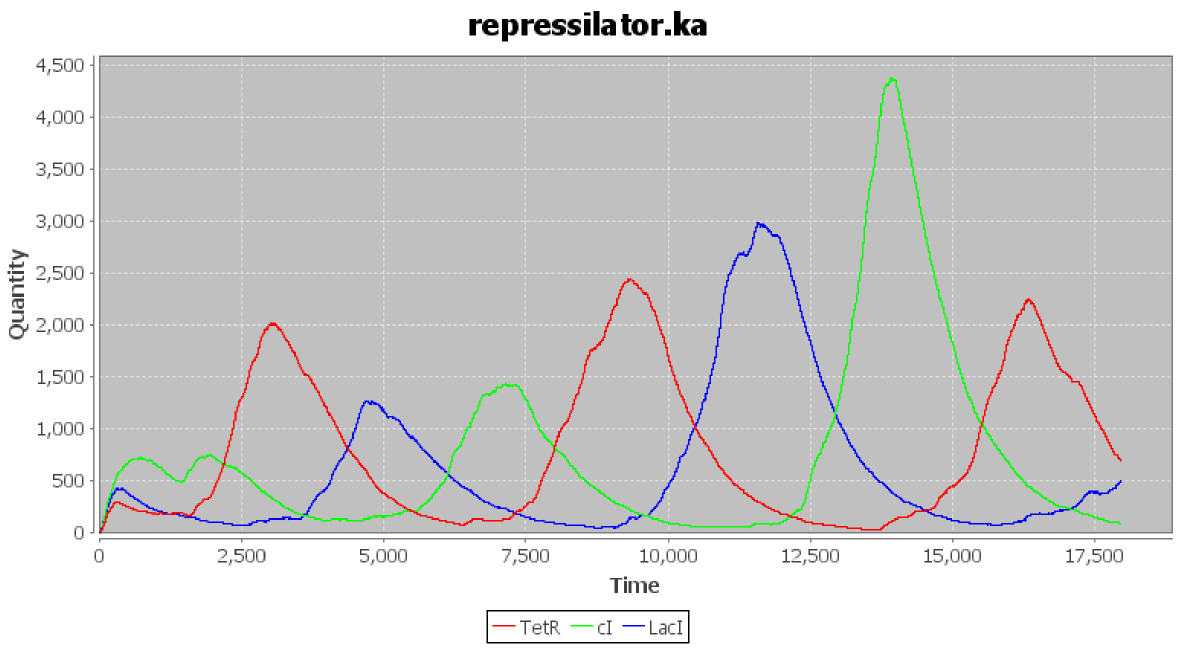


Figure 3.5: Simulation results for the Elowitz repressilator modelled using the Kappa BioBrick Framework. The system stutters to begin with due to the initial conditions of the model favouring none of the three repressors, but as soon as one begins to assert dominance the system falls into its familiar oscillating pattern.

The repressilator combines three simple synthetic genes connected in a loop (Figure 3.4), such that the product of each gene represses the next gene in the loop and is in turn repressed by the product of the previous gene. The output is oscillatory as the repression of one protein by the second allows the third to build up to repress the second, which in turn allows the first to accumulate to repress the third, and so forth (Figure 3.5).

Our model consists of seven types of agents (DNA, RNA, Ribosome, RNAP, and the three repressor proteins involved) and 61 rules. The agents and rules are composed as described previously for eight BioBrick parts: three protein coding sequences (LacI, TetR, and λ-CI), their corresponding promoters, and one each of a terminator and an RBS. Each promoter is designed to be cooperatively bound by up to two of its associated transcription factors (in this case a repressor), and thus is modelled as a concatenation of four DNA agents (two dedicated to repressor binding, one for RNAP binding, and a linker or spacer separating the promoter from any preceding part). Every other BioBrick part is modelled as a single DNA agent. The terminator and RBS are reused across all three devices.

In the paragraphs to follow, we describe the rules of the model according to the BioBrick parts they are associated to. Readers may find Table 3.1 useful in keeping track of the parts and their associated rules.

The first three rules of the model are RNAP and ribosome 'falloff' rules that apply to all DNA agents (BioBrick parts), and an equally universal RNA degradation rule.

Each promoter requires up to $2n\ (2^{n-1}) + 2^n + 2$ rules, where n is the number of transcription factor binding sites, (2^{n-1}) represents the number of occupancy contexts in which a binding or unbinding can occur, and 2^n is the number of transcription factor binding configurations that the RNAP may depend on:

- up to $n\ (2^{n-1})$ rules to describe the possibly cooperative binding of the transcription factor(s)
- up to $n\ (2^{n-1})$ rules more to describe unbinding
- up to 2^n rules to describe the binding of the RNAP
- a singleton rule describing initiation of the transcription process
- a further singleton rule to deal with transcription readthrough by transporting any RNAP that arrives at the linker to the RNAP binding site (thus allowing the action of further rules)

In this model, $n = 2$ for all promoters, hence each promoter requires 14 rules to describe its behaviour. All of the transcription factor binding and unbinding rules are combinatorial in the number of binding sites, in this case two, since (for example) the binding of a transcription factor at one site is conditional on whether or not the other site is already bound. Similarly, we model four rules to describe the binding of RNAP: one for when no transcription factors are bound, one each for when either of the sites are bound, and one for when both sites are bound. The eight rules describing the binding of various agents to the part are all variations on the 'docking' base rule, the four rules describing unbinding are variants on the 'falloff' base rule, and both transcription initiation and transcription walkthrough are based on the 'sliding' base rule.

Of course, if the combinatorics of the promoter are known not to not fully affect its activity, for example the rate of RNAP binding, enumerating all occupancy contexts is possibly an overly detailed solution. Currently it is unclear how one might benefit from the potential regularities in promoter structure by saving on the cost of describing them.

The protein coding sequences each require three rules, all based on the 'sliding' template:

- a rule for its transcription
- a rule for initiating translation
- a rule for the actual translation of the protein

As may be surmised from the rule descriptions, only the first rule is active at the transcriptional level; the other two operate at the translational level.

The RBS is described via two rules:

- a transcription rule ('sliding')
- a ribosome binding rule ('docking')

Transcription of the RBS DNA agent creates an equivalent RNA agent, upon which the ribosome binding rule may then fire as the first step of the translation process.

The terminator also requires two rules:

- a 'falloff' rule with enhanced rate to represent its efficiency at terminating transcription
- transcription readthrough ('sliding') if its terminator function fails

The first of these terminator rules trumps the generic RNAP 'falloff' rule, thus making the terminator much more efficient at removing RNA polymerase from the DNA chain than any accidental falloff.

The proteins created in the model act as transcription factors as described in the promoter rules, but have no other activity beyond three rules describing their degradation. These final three rules are the only rules in the model not specified by the Kappa BioBrick Framework, but are required to produce the oscillations characteristic of the repressilator.

The initial conditions of the model contain one of each of the BioBrick devices shown in Figure 3.4, as well as a population of RNA polymerase and ribosomes. By identifying the LacI, TetR, and λ-CI proteins as the observables of interest, simulation of the agents and rules described above generates a time course similar to that shown in Figure 3.5.

Note, of course, that the framework as described above is not the only way to model BioBrick parts in Kappa; other, less modular formalisms may be of equal use to the modeller. Furthermore, by no means does it completely capture every interaction necessary in fine detail. As an example, RNA degradation process makes no distinction between exo- and endonucleases; it simply destroys RNA with free binding and downstream sites, which is roughly equivalent to half the activity of an exonuclease. A simple modification to the base framework would be to differentiate between the two. As another example, the transcription factors in the framework associate specifically with their binding site upon the promoter, whereas in reality they search for their appropriate binding sites by 'sliding' along the DNA chain; a possible extension of the framework would be to take this action into account. A further extension might be to make use of the linker portion of the promoter and the promoter's transcription readthrough rule to represent the length of non-coding region between BioBrick devices. We might even refine the RNA agent to model the ability for multiple Ribosomes to exist on a single mRNA, thus fully distinguishing between PoPS and RiPS.

Current levels of implementation allow the skeleton of the framework -- the unrefined 'docking', 'sliding', and 'falloff' rules described above -- to be automatically generated via LMS-Kappa. Given a suitably detailed source, such as an ideally populated Registry of Standard Biological Parts, one can imagine automating the modelling of such aspects as promoter structure (repressor or activator, number of transcription factor binding sites and their cooperativity if any) and rate information (transcription factor affinity, RBS and terminator efficiency, PoPS, and RiPS). However, we envisage that the modeller will always need to provide detailed protein interactions outside the scope of the framework so as to maintain the flexibility required of them.

These protein interactions are the subject of the next section, where we build upon the repressilator just described.

4 Extending the Repressilator: Light-Mediated Synchronisation

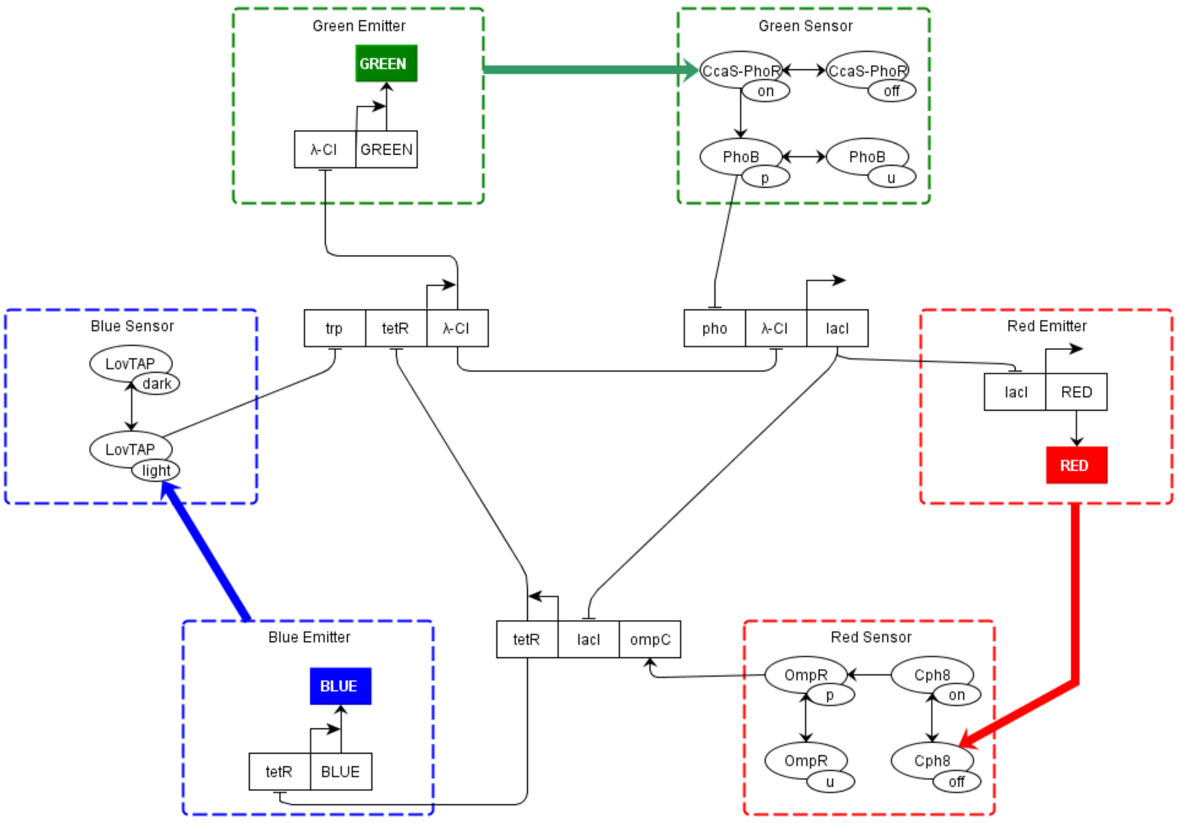


Figure 4.1: Modelling light emitting and light sensing pathways coupled to an Elowitz repressilator. At the centre the oscillating repressilator regulates the emission of light in the system. The light sensing pathways then provide input to this central regulator via a second promoter coupled to the same coding sequence, reinforcing or adjusting responses to synchronise the system. The central repressilator and the light emitting pathways are genetic components, built solely around the Kappa BioBrick Framework, but the light sensing pathways are protein interaction components that must need be manually specified by the modeller (see Figure 4.2). [Reprinted from (17) with permission from Elsevier.]

The Kappa BioBrick Framework also provides a basis for modelling component devices that combine into a larger and more complex system. Individual modules may be developed and verified independently, simplifying the process of breaking down the overall effort into manageable units amenable to repeated cycles of refinement and extension. This is accomplished in a similar manner to the modularity of the individual BioBrick parts -- individual devices, consisting of a set of agents, rules, and associated variables, are simply concatenated together into a single larger model. When composing in this manner, care must be taken that a specific biological entity (an agent and its sites) is depicted in the same way across different modules to ensure that it is shared as intended.

In addition, depending on the level of granularity desired, further rules may be used to describe the protein interactions necessary for the model (for example the activity of a signalling pathway linking a translation product to a transcription factor). These may be safely added independently of the Kappa BioBrick Framework and thought of as a modular component themselves. We proceed to elaborate on these points in the paragraphs to follow.

One of the weaknesses of the Elowitz repressilator is that without external regulation the system oscillations are extremely imprecise and do not last over time, so that the oscillations of multiple cells each running their own version of the repressilator rapidly desynchronise or die off. To address this problem, the 2010 Edinburgh iGEM team designed a light-mediated communication system based on the Elowitz repressilator (Figure 4.1), involving the establishment of three independent channels of communication in different spectral wavelengths **(17)**. The goal of the project was to develop a multi-cellular system capable of self-reinforcing collective synchronisation, with the eventual aim of enabling bacterial populations to interact with each other as well as with purely electronic systems via light.

As a first step to achieving the above, each gene product in the repressilator loop is used to repress the production of light of a particular wavelength. In other words, each wavelength is associated with the lack of an associated repressor (red light with lack of LacI, blue light with lack of TetR, and green light with lack of λ-CI).

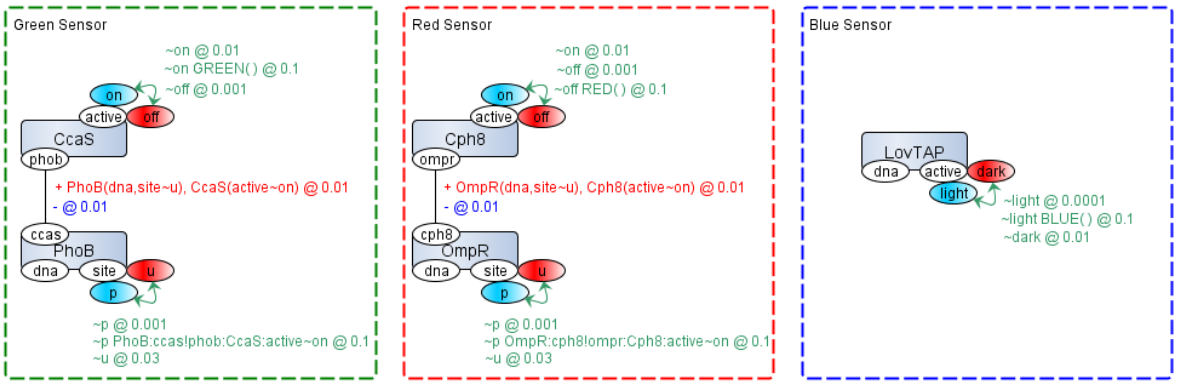


Figure 4.2: Visualisation of the light sensing pathways in the light-mediated repressilator. The red and green sensors are both two-component regulatory systems that induce changes in conformation when subjected to light of the appropriate wavelength. The green sensor is usually 'off', but when subjected to green light activates CcaS, which binds to and phosphorylates PhoB, which then acts as a transcriptional repressor. In contrast, the red sensor is usually 'on' and activating a downstream promoter; when red light is sensed, Cph8 is inactivated, which stops binding to OmpR and thus encourages its dephosphorylation, which in turn prevents it from acting as a transcriptional activator. In summary, incoming green light activates a repressor, as opposed to incoming red light which deactivates an activator instead; both thus have a symmetrical inhibitory effect on their coupled protein in the core repressilator. The blue light sensor is an allosteric hybrid protein with a simple single-component mechanism: activation via blue light switches LovTAP to its 'light' configuration, whereupon it is free to act as an inhibitor upon the core repressilator.

In the meantime, the light sensing pathways (Figure 4.2) provide input to the core repressilator to reinforce the oscillatory response. The model assumes that they are constitutively expressed, unlike the proteins generated by the core repressilator. For each of the three repressilator proteins, a second promoter is coupled from the light sensing pathway to the original promoter, such that inhibition from either repressor is sufficient to inhibit its production; this assumption is based on combinatorial promoter characterisation in Cox et al. **(18)**. The activated green and blue sensors explicitly inhibit LacI and λ-CI respectively, while the activated red sensor ceases promotion of TetR, hence inhibiting it implicitly.

If the effect of the light emitting pathways is to 'broadcast' the current state of the host's repressilator to its neighbours, the proposed effect of the light sensing pathways is to 'adjust' its state to match those of its neighbours instead. When activated, ideally the sensors either reinforce the current state of the repressilator in the host (because it is already synchronised with its neighbours) or modify it to bring it closer to the desired behaviour. This communication occurs at a much faster rate than the transcriptional interactions of the core repressilator, but it is unclear at this stage (when approaching this system guided only by intuition) whether or not this synchronisation will actually occur if implemented *in vivo*.

Modelling the light sensors and emitters in conjunction with the core repressilator, and analysis of the proposed system as a whole, is thus crucial to understanding it. This model of light communication is composed from seven components (the core repressilator, along with emitters and sensors for the three wavelengths), each of which we may consider a functional model in its own right. Each of these could then be validated individually before being combined into a single host, and then tested in both communicating and non-communicating colonies.

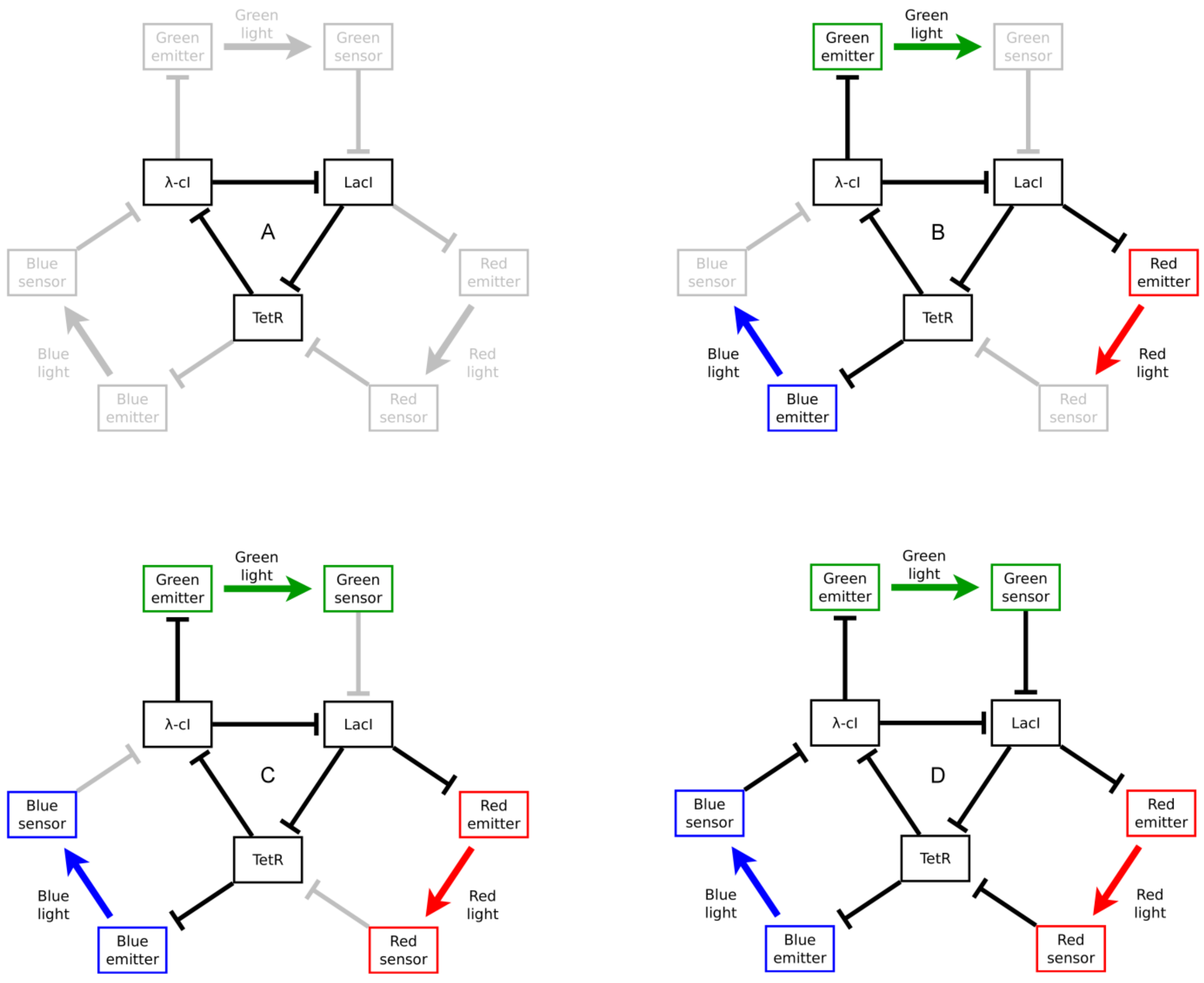


Figure 4.3: Iterative development of the light communication system, from core repressilator to complete network, showing the modular nature of the system. A) Core repressilator. B) Light emission pathways. C) Light sensing pathways. D) The complete network. [Adapted from (17) with permission from Elsevier.]

The first iteration of development adopts the core Elowitz repressilator component (Figure 4.3a) described previously in Section 3. We then add the light emitting components (Figure 4.3b) to the model, producing oscillating light outputs linked to the oscillations of the core repressilator shown in Figure 3.4. Biologically, these components were all synthetically created to purpose in the iGEM project: green light emissions based on the standard firefly (*Photinus pyralis*) luciferase enzyme which then underwent site-directed mutagenesis to create red light, while blue light emissions were developed from a bacterial luciferase from *Xenorhabdus luminescens*. Both the central repressilator and the light emitting components are formed solely from the BioBrick parts -- the proteins produced and involved have no function other than as a transcription factor, and thus there is no necessity to manually model protein-protein interactions.

The next development iterations focus on the light sensing pathways. Similar to the light emission pathways, these were biologically engineered to purpose: the red light sensor was based on a bacterial phytochrome, the blue light sensor on a plant phototropin, and a novel fusion protein was designed as a green light sensor. Unlike the core repressilator and light emitting components, the proteins involved in the light sensing pathways are assumed to be constitutively expressed in the cell. Thus they are modelled purely on the protein interaction level without any use of the Kappa BioBrick Framework. By not allowing the produced transcription factors to bind to the repressilator (Figure 4.3c), the response of the individual light-sensing pathways may be tested via perturbation analysis. Only when satisfied with their performance need this restriction be lifted (Figure 4.3d), resulting in the complete model of the synthetic circuit.

Usually throughout the development of a model, rules must be refined and their accompanying kinetic parameters tuned to maintain the desired behaviour (in this example, oscillations). Due to constraints on time and equipment, these rate parameters may be originally derived from *in silico* trial-and-error analysis rather than *in vivo* experimentation. The current version of KaSim also allows for the declaration of global variables that can be used to control the rates of multiple rules, thus affording another layer of modularity in the development of the model since rules with similar function (for example, readthrough transcription) may be controlled with fewer parameters. The latest version of the light-mediated repressilator model provided to the reader on RuleBase (http://rulebase.org) makes use of this capability.

Up to this point we have simply tied the BioBrick parts comprising the repressilator to the protein interactions of three simple signalling pathways. The intracellular model may now be extended further to simulate the behaviour of an idealised virtual colony of bacteria communicating with each other using the light produced within each cell. This may be done by utilising the simplifying assumption that the bacteria are non-motile and closely packed in a two-dimensional hexagonal biofilm. Additional rules to the model represent the communication of light between neighbouring cells, each responsible for maintaining its own repressilator. A snapshot of a sample simulation is shown in Figure 4.4, with isolated non-communicating cells shown on the left and communicating cells on the right.

The light levels in each cell are recorded at each sample point during the simulation, along with the colony mean light levels and the standard deviation of the individual cell light levels from these colony means. Accurate average behaviour is recorded by measuring results (Figure 4.5) over sufficiently long simulations. These results show that a communicating colony has less time-averaged deviation in light levels between cells, and therefore increased synchronisation.

So far in this chapter, we have explored a number of Kappa models in synthetic biology, from those based on protein-protein interactions to the purely genetic. We have attempted to highlight both the inherent strengths of adopting a rule-based approach to modelling, and the manner in which the Kappa BioBrick Framework structures the modelling of BioBrick parts whilst leaving room for extension and refinement. To conclude from here, we take a look at the future of rule-based modelling in Kappa, paying particular attention to the following questions: what is missing from the methods described above, and how we may fully exploit our advantages.

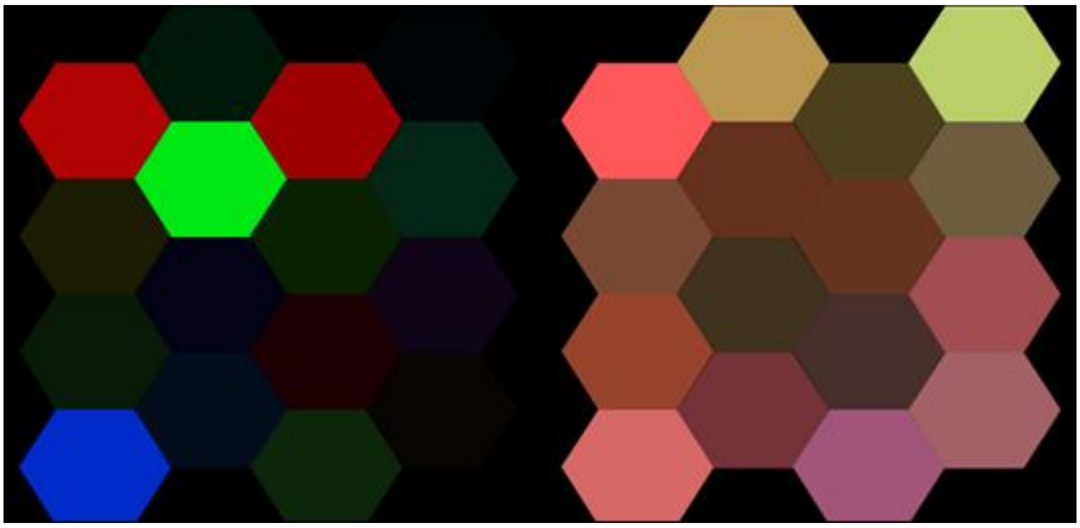


Figure 4.4: A sample simulation snapshot of the multicellular light-mediated repressilator model. Isolated cells (non-communicating) are shown on the left, each running a separate repressilator and broadcasting a clearly defined output wavelength independently of its neighbours. The communicating cells on the right show the effect of light-mediated synchronisation on the overall state of each cell: their output wavelengths are muddier but more closely correlated.

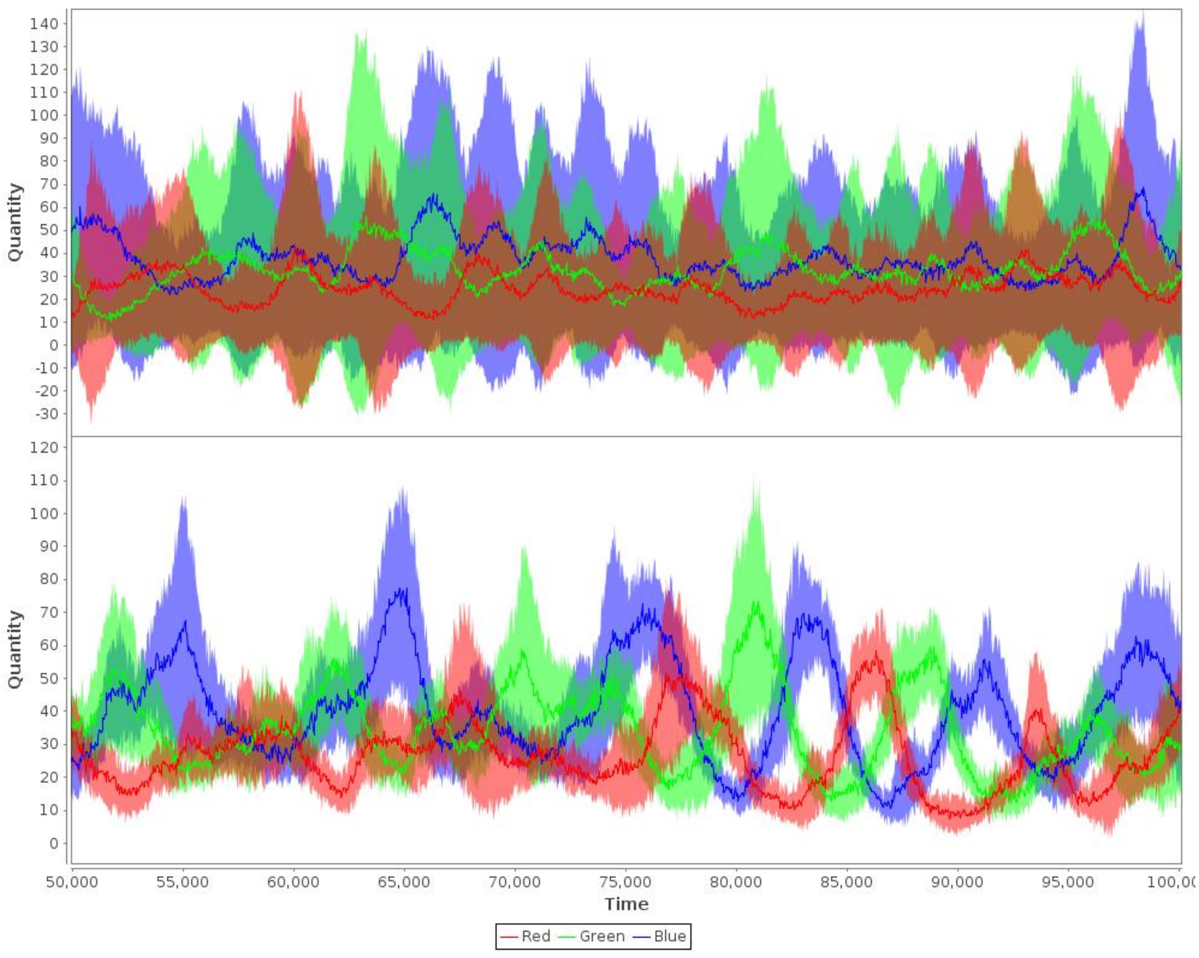


Figure 4.5: Comparison of mean cell light levels in 4x4 colonies of cells, both isolated (top) and communicating (bottom) between cells in the virtual colony. Shaded areas on the graphs show standard deviation of cell light levels from colony mean. Communicating colony shows reduced time-averaged standard deviation (R:9.46, G:11.76, B:13.49) compared to the isolated colony (R:25.71, G:36.35, B:39.44), and hence increased coherence in cell activity levels. Units (time and quantity) are arbitrary. [Adapted from (17) with permission from Elsevier.]

5 Looking to the Future

A fundamental challenge to synthetic biology is the engineering of biological parts with behaviour that is well-defined in relation to other parts. This not only requires controlled and precise measurement protocols, but also a modelling language for the formalisation of these interactions. The Kappa rule-based biological modelling language provides a means to this end -- a set of rules describing the ways in which biological entities interact with other entities present in the system at a tunable level of detail.

The rule-based approach embodied by languages such as Kappa has a number of advantages over its alternatives, not least in the fact that it greatly reduces the combinatorial complexity of the system description in comparison to more traditional reaction-based models. Modular rules describe the functionality of individual BioBrick parts in an easy-to-understand manner, thus aiding comprehension of the underlying biology, and can be easily reused both in different contexts within a model and across multiple different models. The associated Kappa BioBrick Framework is well-suited to working with individual parts and this can only improve in the future with the development of dedicated support tools and databases. The modularity of the approach also makes it easy to apply an iterative development methodology to the problem, easing the process of building, analysing, and understanding the model.

On the other hand, not all models can be constructed automatically from genetic components in the form of BioBrick parts, and the modeller input necessary to fill in this knowledge gap is non-trivial and may daunt a newcomer to the language. Furthermore, the framework itself is neither as readable nor as cleanly modular as we may wish. To address the second of these we turn to a pair of extensions to the Kappa language (their application in this manner first proposed by Elaine Murphy): meta-Kappa **(19)** and Coloured Kappa.

Meta-Kappa introduces agent hierarchies in which children may inherit sites from their parents and may add new sites or replicate existing ones as well. Rules may be written with both parents and children, but all agents must be concretely specified (at the bottom level of the hierarchy) in the initial conditions of the model. The use of meta-Kappa is especially useful in concisely representing promoters with multiple binding sites for transcription factors.

Coloured Kappa improves the modularity of rules reliant on the BioBrick part number (for example, the transcription rule depicted in Figure 3.1) by allowing rules to contain variables, and rates of such a rule to be a function of said variables. Without Coloured Kappa, we are forced to write a transcription rule for every BioBrick part in the system; with Coloured Kappa, we can simply write a single rule conditioned on the BioBrick part number as a variable.

Both meta-Kappa and Coloured Kappa have value in simplifying the manual user input to the Kappa BioBrick Framework. However, neither of them have yet been integrated with the framework, and it would be useful to accomplish this in the future.

Another extension to Kappa of value to the framework is Thermodynamic Kappa **(20)**. Thermodynamic Kappa helps us better express the relationship between forward and backward kinetic rates (i.e. the equilibrium constant), such as that given by the different affinities between the transcription and translation machinery with various portions of the DNA and RNA. These affinities realise the difference in potential energies between the various configurations. In Thermodynamic Kappa we can state these directly using energy patterns, that is, a list of connected agents with their associated energies. In the repressilator model this could be used to express the differences in the 'docking', 'sliding', and 'falloff' equilibrium constants for the various DNA-RNAP or RNA-Ribosome complexes. For instance, by declaring that the energy for RNAP bound to a DNA terminator sequence is higher than that of RNAP bound to any other DNA part, we may implicitly state that RNAP dissociates from DNA more readily when sitting on top of a terminator. In the same way, cooperativity among transcription factors when binding to DNA can be described by a lower energy for the configuration in which two transcription factors are bound to DNA than when one of them is bound without the other.

An implementation of Thermodynamic Kappa based on the Metropolis algorithm **(20)** is available in LMS-Kappa. Further work is in progress to implement meta-Kappa and other extensions of Kappa not mentioned in this chapter on top of the LMS-Kappa core. Most importantly for the subject of this chapter, a preliminary implementation of the Kappa BioBrick Framework is also available as an extension in the standard distribution of LMS-Kappa (https://github.com/sstucki/lms-kappa). The extensibility of LMS-Kappa will eventually allow all of these language extensions to be adopted by the modeller according to the needs of the model, thus greatly contributing to even more flexible methodologies for modelling synthetic biology in Kappa.

What else can we think of for the future of Kappa modelling in synthetic biology? One possible step forward could be to extend the prokaryotic Kappa BioBrick Framework for eukaryotic purposes, allowing the strengths of Kappa to come into play for fields such as RNA-focused synthetic biology. We have already discussed in Section 3 the refinement of elements of the existing framework to better reflect known biology (for example the better models of RNA degradation) or to take better advantage of the benefits of Kappa in alleviating combinatorial complexity (for example in modelling the search process that occurs when transcription factors search for a suitable promoter binding site).

Finally, an exciting new domain of application is the concept of whole-cell modelling as exemplified by Karr et al. **(21)**. Whole-cell models combine multiple modelling approaches, such as ordinary differential equations and flux balance analysis, to integrate all of a cell's molecular components and their interactions in a single computational object. As applied to synthetic biology, we might begin by proposing a virtual chassis, which would allow us to test *in silico* how proposed BioBrick devices (modelled for example using the Kappa BioBrick Framework) affect the physiology of the host. A grander vision would then be to have a selection of not only different chassis (for example, yeast, *Escherichia coli*, *Bacillus subtilis*, and so forth), but also different versions of chassis components (for example, signalling pathways) that would allow potential modellers to customise their model via plug and play.

Another application of the whole-cell modelling paradigm would be to test the Kappa BioBrick Framework's ability to account for resources (RNA polymerases and ribosomes) shared with other processes extant within the host. As stated throughout this chapter, Kappa allows for the refined modelling of transcriptional and translational activity, for instance facilitated search by transcription factors and multiple ribosomes upon mRNA. In a whole-cell model similar to that implemented by Karr et al., we might on the one hand use other modelling techniques such as flux balance analysis for metabolism, and delegate the interactions of DNA and RNA to a high resolution Kappa model. This would allow us to apply the best modelling techniques to their most appropriate usage, and tie them all together in a coherent whole.

To conclude this chapter, we would like to reiterate the concepts we have covered thus far. We have introduced the rule-based modelling language Kappa via a series of four examples: the first kinase-phosphatase model to acquaint readers with the basic Kappa language and visualisations thereof; the second to repeat this in a synthetic biological example of a unidirectional 'on' switch to make concepts more concrete; the third to introduce the Kappa BioBrick Framework via the repressilator; and the fourth to extend this repressilator with protein-based light-mediated communication. We have also briefly introduced recent developments in the Kappa language, such as LMS-Kappa, aimed at implementing and extending the usability of the Kappa BioBrick Framework.

Of course, we should warn the reader that part of the excitement of synthetic biology is that it is a field in flux; therefore the methodology which we propose should itself be thought of as an open-ended process, more as a series of guidelines, than a closed standard. On the other hand, we hope that throughout this chapter we have convinced the reader that Kappa and the Kappa BioBrick Framework can be of help to a modeller with a computational background in understanding synthetic biology, thus providing fresh insight into this nascent and exciting discipline.

References

[1] Danos, Vincent, and Cosimo Laneve. "Formal molecular biology." *Theoretical Computer Science* 325.1 (2004): 69-110.

[2] Blinov, Michael L., et al. "BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains." *Bioinformatics* 20.17 (2004): 3289-3291.

[3] Bachman, John A., and Peter Sorger. "New approaches to modeling complex biochemistry." *Nature Methods* 8.2 (2011): 130.

[4] Gillespie, Daniel T. "Exact stochastic simulation of coupled chemical reactions." *The Journal of Physical Chemistry* 81.25 (1977): 2340-2361.

[5] Feret, Jerome, and Jean Krivine. "KaSim3 Reference Manual." (2013). https://github.com/jkrivine/KaSim/blob/master/man/KaSim_manual.pdf?raw=true. Accessed 26 June 2013.

[6] Danos, Vincent, et al. "KaSpace: a language for the combinatorial assembly of biological complexes." (2013) [To Appear]

[7] Pedersen, Michael, et al. "A High-Level Language for Rule-Based Modelling." (2013) [To Appear]

[8] Goldbeter, Albert, and Daniel E. Koshland. "An amplified sensitivity arising from covalent modification in biological systems." *Proceedings of the National Academy of Sciences* 78.11 (1981): 6840-6844.

[9] Wang, Dong, et al. "A genomic approach to identify regulatory nodes in the transcriptional network of systemic acquired resistance in plants." *PLoS Pathogens* 2, no. 11 (2006): e123.

[10] Moore, John W.. "Foundation technologies in synthetic biology: tools for use in understanding plant immunity." PhD Thesis, University of Edinburgh (2012).

[11] Gardner, Timothy S., et al. "Construction of a genetic toggle switch in *Escherichia coli*." *Nature* 403.6767 (2000): 339-342.

[12] Shetty, Reshma P., et al. "Engineering BioBrick vectors from BioBrick parts." *Journal of Biological Engineering* 2.1 (2008): 1-12.

[13] Marchisio, Mario A., and Jorg Stelling. "Computational design of synthetic gene circuits with composable parts." *Bioinformatics* 24.17 (2008): 1903-1910.

[14] Chandran, Deepak, et al. "TinkerCell: modular CAD tool for synthetic biology." *Journal of Biological Engineering* 3.1 (2009): 19.

[15] Marchisio, Mario A., et al. "Modular, rule-based modeling for the design of eukaryotic synthetic gene circuits." *BMC Systems Biology* 7.1 (2013): 42.

[16] Elowitz, Michael B., and Stanislas Leibler. "A synthetic oscillatory network of transcriptional regulators." *Nature* 403.6767 (2000): 335-338.

[17] Stewart, Donal, and John R. Wilson-Kanamori. "Modular Modelling in Synthetic Biology: Light-Based Communication in *E. coli*." *Electronic Notes in Theoretical Computer Science* 277 (2011): 77-87.

[18] Cox, Robert S., et al. "Programming gene expression with combinatorial promoters." *Molecular Systems Biology* 3.1 (2007).

[19] Danos, Vincent, et al. "Rule-based modelling and model perturbation." *Transactions on Computational Systems Biology XI*. (2009): 116-137.

[20] Danos, Vincent, et al. "Thermodynamic Graph-Rewriting." *Lecture Notes in Computer Science* 8052 (2013): 380-394.

[21] Metropolis, Nicholas, et al. "Equation of state calculations by fast computing machines." *The Journal of Chemical Physics* 21 (1953): 1087.

[22] Karr, Jonathan R., et al. "A whole-cell computational model predicts phenotype from genotype." *Cell* 150.2 (2012): 389-401.